VTA++: Expanded Design Space Exploration with an Enhanced Versatile Tensor Accelerator

Suvadeep Banerjee, Steve Burns, Pasquale Cocchini, Abhijit Davare, Shweta Jain, Desmond Kirkpatrick, Anton Sorokin, Jin Yang, Zhenkun Yang

Intel Labs
Motivation: Design Methodology Research

- Inspired by 2018 Turing Award paper [1]:
  - Domain-specific languages (DSLs) and architectures (DSAs) are key.
  - Gains “… will require a vertically integrated design team that understands applications, domain-specific languages and related compiler technology, computer architecture and organization, and the underlying implementation technology.”

- Research Goal
  - Lower the design cost for DSLs deployed onto DSAs

- Hypothesis
  - Incremental feature addition with a vertical development stack
  - Neither software-first nor hardware-first design

- TVM/VTA was an appealing starting point
  - Development stack spans workload down to hardware, yet fast simulation possible
  - User-schedulable compiler provides rich software choices
  - Parameterized hardware presents a large design space

Goals: Increase size of the VTA design space and enable higher performance

Outline

- Background: VTA
- Increasing Throughput
- Expanded Design Space
- Results
- Conclusion
Background: Versatile Tensor Accelerator (VTA) [1]

- **DNN Inference Accelerator**
  - 8-bit input/weight, 32-bit acc
  - GEMM and ALU units
- **Multiple targets**
  - **fsim**: behavioral, C++
  - **tsim**: cycle-accurate, CHISEL
  - Others: pynq, de10, focl, etc
- **CHISEL** [3]
  - Hardware construction language which produces RTL

---


- GEMM datapath in tsim was pipelined, but control logic was not
- Added pipelined control (index generator) to reduce initiation interval to 1 cycle, resulting in a ~4x throughput increase
- Similar changes for ALU, but acc src/dest port restriction
- Optional flops added to meet frequency targets for larger GEMM shapes
Added configurable memory width (8-64 bytes)
Now allow multiple outstanding transactions with out-of-order completion
Expanded Design Space: GEMM and scratchpads

- **GEMM shape**
  - Separate BLOCK_IN from BLOCK_OUT
  - Add BATCH support

- **Larger scratchpads**
  - Also permit 64-bit uops
  - Retain 128-bit instruction width: expand some fields while shrinking others

Expanded Design Space: Compiler

- **Tiling Parameter Search**
  - Heuristic guided analytical scheme for mapping conv2D and depthwise conv2d to VTA configurations
    - Simple memory model allows us to analytically estimate cycle count impact of scheduling decisions
    - With vastly more configurations, measurement is expensive
  - VTA previously used a measurement-guided database of optimal schedules per configuration from AutoTVM

- **Double buffering improvements**
  - Previous scheme already allowed overlap in load/compute/store execution
  - Now enhanced to allow greater reuse of scratchpad data
Additional Layers

- Depthwise-Conv enables Mobilenet 1.0 [1]
  - Elementwise 8-bit multiplication instruction added
- Pooling support allows FC layer in ResNets [2] to be offloaded to VTA as well
  - Max Pooling
    - Load instruction augmented with parameterized pad values: 0 and MAX_NEG
  - Average Pooling
    - Approximate division using ALU shifts and adds

Results: Cycle count vs. Scaled area

- Scaled area is for an ASIC process
  - Using split register files for larger configurations
- GEMM/ALU pipelining
  - ~4x reduction in cycle count
  - Minimal area change
- Must balance compute vs. scratchpad/bandwidth
- Not shown here:
  - Batch size > 1
  - Rectangular GEMM shapes
**Results: Roofline Analysis**

- Roofline plot [1] identifies compute and communication bottlenecks
  - Compute: y-axis, GEMM MAC count
  - Communication: diagonal, bytes/cycle of memory bandwidth
- Each point represents a ResNet-18 RTL simulation
- Much closer to theoretical maximums
  - Lower MACs are compute bound,
  - Higher MACs are memory bound

---

Results: Continuous Integration

- **Test-Driven design (TDD) loop** consists of CHISEL unit tests
  - Initially used for development
  - Valuable for hardware coverage
- **Runtimes shown for ResNet-18 workload**
- **Cycle-accurate tsim simulation is surprisingly fast**
  - Most VTA features can be simulated in under 3.5 minutes
  - Parallelization not restricted by physical devices or licensing
- **Behavioral simulation helps debug**
  - Dynamic Trace-Based Validation
Conclusion

- Enhanced performance and expanded design space
- Open source contributions:
  - Pipelined GEMM, pipelined ALU, and memory interface changes have been upstreamed to the tvm-vta repository
  - All other changes are available in a fork
- Read our arXiv paper to learn more:
- Our research focus is improving design methodologies
  - We are hiring!
- Contact me with comments/questions at firstname.lastname@intel.com